EAGLE-200 Uploader API Specification

Version 2.5

January 8, 2020

Trademarks

Notice

Uploader API Specification

Table of Contents

1. Overview

The Rainforest EAGLE-200 gateway can “push” meter and device data automatically to a cloud server on the Internet. This document specifies the various configuration options available as well as the API that is used for the upload.

1.1. Configuration

1.1.1. Modes of Operation

The Uploader has 2 modes of operation:

1. Streaming: Data is streamed by the gateway as it is received from the devices and is not stored. This can be enabled by setting UploadSize to 0.
2. Buffered upload: Data is uploaded only after certain, configurable parameters have been reached. In this case, data packets are collected until either a configurable number of packets has been received or some timeout has been reached. If the upload site is not available, data packets will be stored within the Rainforest device until the upload site can be reached again (UploadSize must be non-zero for data packets to be stored).

1.1.2. Configuration Files

There is a configuration file for each known upload site. The files are in XML format and can be configured using the EAGLE-200 REST API. See the Uploader Configuration section of this document for details on how to create and modify the configuration file.

Here is a sample configuration file:

```
<UploadSite>
 <Provider>simple_streaming</Provider>
 <Description>RFA staging server</Description>
 <HostName>api.rainforestautomation.com</HostName>
 <User>0033e4</User>
 <Port>443</Port>
 <Password>password here</Password>
 <Protocol>https</Protocol>
 <Url>/post_data</Url>
 <Format>RFA</Format>
 <UploadSize>20</UploadSize>
 <UploadPeriod>600</UploadPeriod>
</UploadSite>
```

The elements of the Configuration File are:

Element	Default	Description
Provider		Simple uploader name (i.e. "Rainforest"). Must be unique to this EAGLE-200.
Description		Long description of the upload site.
Protocol	http	http or https
Hostname		Server identifier to which we are uploading
Url		Path part of URL (appended to the Hostname to form the complete URL)
Port	80 (http) 443 (https)	Port number used by uploader
User		Username, if required by the destination server (Optional)
Password		Password, if required by the destination server (Optional)
Format		• RFA – JSON format • XML:Raw – unprocessed XML from the Zigbee module • XML:Simple – processed XML in hierarchical format
UploadSize	0	Size of upload packets. When this number of samples has been reached, a batch upload will be initiated. Setting this to "0" will start Streaming mode.
UploadPeriod	0	Frequency of upload packets. When this number of seconds has elapsed, a batch upload of all outstanding samples will be initiated. Setting this to "0" will start real-time uploading. This element overrides UploadSize.
X-Header1		Custom header #1 (if required).
X-Header2		Custom header #2 (if required).
X-Header3		Custom header #3 (if required).

2. Data Types

Here is a list of the various types of data which can be uploaded via the Generic Cloud Uploader.

Item	**Units**	**Description**
Demand	kW	Instantaneous demand
Summation	kWh	Cumulative summation delivered & received
Message	Text	Text message
Price	Currency	Price per kWh of electricity

3. Data Formats

3.1. XML Raw Format

3.1.1. Request

The EAGLE-200 sends data in a POST request. POST requests have the following structure:

```
POST <URL> HTTP/1.1
<headers>
<blank>
<body>
```

Where:

- Every line ends with the newline character (0x0A).
- `<URL>` is the Uniform Resource Locator (web address) of the external web server.
- `<headers>` are a variable number of HTTP headers; each header is on its own line.
- `<blank>` is a blank line, consisting only of the newline character (0x0A).
- `<body>` is the main text of the POST request, which has the structure shown below.

The body of the POST request has the following structure:

```
<?xml version="1.0"?><rainforest timestamp="0000000000s"
version="2.0"   macId="0xffffffffffff">
     <body>
</rainforest>
```

The first line is the standard XML (eXtensible Markup Language) Prolog. This is followed by the Root Element, which for our purposes is enclosed by the tags `<rainforest>` and `</rainforest>`. The Root Element has three attributes:

- timestamp -- this is an integer with a standard Unix timestamp, i.e., number of seconds since Jan.1, 1970.
- version – this is for future use and is currently undefined.
- macId – this is a 12-digit hexadecimal number. It is the Ethernet MAC Address of the EAGLE-200.

The body of the Root Element consists of XML Fragments. An XML Fragment is a stripped down XML Element. The EAGLE-200 uses XML Fragments to simplify the parsing of the data stream, while providing a data structure that is flexible and human readable.

The XML Fragments have the following structure:

```
<tag>
     <element>value</element>
     …
</tag>
```

Where:

- Every line ends with the newline character (0x0A).
- `<tag>` is the start tag for the XML Fragment; each type will have a unique tag name;
- `<element>` is the start tag for an element; there will be one or more child elements in the fragment; each element will have a unique element name.
- … indicates the variable number of specific elements

Element values can be of various types:

- `{string}` indicates an element consisting of Extended ASCII text
- `{enumeration}` indicates an element that can have a specific list of values.
- `0xFFFFF` indicates an element consisting of a base16 (hex) number
- `00` indicates an element consisting of an integer
- `000.000` indicates an element consisting of a signed decimal number

`value1|value2|value3` – vertical bars separate valid values in an enumeration list.

Note that element names are case insensitive; the case is used strictly for legibility. XML parsers should be designed to ignore case when receiving requests from the EAGLE-200.

Example

Here is an example of a POST request:

```
POST sample.url HTTP/1.0
Content-type: text/xml
Content-Length: 526

<?xml version="1.0"?>
<rainforest timestamp="0000000000s" version="2.0"
macId="0xffffffffffff">
<InstantaneousDemand>
<DeviceMacId>0xd8d5b9000000b74d</DeviceMacId>
<MeterMacId>0x001d230100402d72</MeterMacId>
<TimeStamp>0x211cc7a8</TimeStamp>
<Demand>0x000032</Demand>
<Multiplier>0x00000001</Multiplier>
<Divisor>0x000003e8</Divisor>
<UnitOfMeasure>0x00</UnitOfMeasure>
<DigitsRight>0x03</DigitsRight>
<DigitsLeft>0x06</DigitsLeft>
<SuppressLeadingZero>Y</SuppressLeadingZero>
<Protocol>Zigbee</Protocol>
</InstantaneousDemand>
</rainForest>
```

Note that every line in the above example actually ends with the newline character (0x0A). This is not shown explicitly for clarity.

3.1.2. Response

The EAGLE-200 expects to see a valid HTTP response to each POST request. These look like:

```
HTTP/1.0 <code>

<headers>

<blank>

<body>
```

Where:

- Every line ends with the newline character (0x0A).
- `<code>` is an HTTP status code, which consists of a 3-digit number and a short text phrase. This is usually “200 OK”.
- `<headers>` are a variable number of HTTP headers; each header is on its own line.
- `<blank>` is a blank line, consisting only of the newline character (0x0A).
- `<body>` is optional and can contain a single XML fragment.

Example

Here is an example of a reply:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Cloud-Trace-Context: e5c0ce6b7f101c54f0714cff8e2b883e
Date: Mon, 12 Aug 2019 22:49:06 GMT
Server: Google Frontend
Content-Length: 2
```

Note that every line in the above example actually ends with the newline character (0x0A). This is not shown explicitly for clarity.

Replies do not normally have a `<body>` component, and simply acknowledge a POST sent by the EAGLE-200.

Note that the server should be configured to send a minimal number of headers in responses. Excessive headers can greatly contribute to the traffic bandwidth.

3.2. XML Simple Format

3.2.1. Request

If the `<Format>` field in the Configuration File is set to `XML:Simple`, then the EAGLE-200 will upload data in XML that has been reformatted into a structured, hierarchical style. In this case, the POST requests are the same as shown above, except the body of the POST requests will have the following structure:

```
<?xml version="1.0"?><rainforest timestamp="1565650146s" version="2.0"
macId="0xd8d5b900a9e8">
<XmlSimple>
 <NetworkInterface>0xd8d5b9000000fca8</NetworkInterface>
 <HardwareAddress>0x00078100005a499f</HardwareAddress>
 <Protocol>Zigbee</Protocol>
 <Manufacturer>Generic</Manufacturer>
 <ModelId>electric_meter</ModelId>
 <TimeStamp>1565646751</TimeStamp>
 <ComponentFixedId>0</ComponentFixedId>
 <MainTag>InstantaneousDemand</MainTag>
 <Variables>
   <Variable>
     <Name>variable1</Name>
     <Value>0.050000</Value>
     <Units>kWh</Units>
   </Variable>

        ...

   <Variable>
     <Name>variableX</Name>
     <Value>0000</Value>
     <Units>min</Units>
</Variables>
</XmlSimple>
</rainforest>
```

Note that the Prolog is the same as the XML Raw format, and the Root Element is still enclosed by the tags `<rainforest>` and `</rainforest>` with the same three attributes (timestamp, version, macId). However, the body of the Root Element is now enclosed by the tags `<XmlSimple>` and `</XmlSimple>` and it has a more structured format.

The XML Simple format consists of a header block, which is common to all types of XML Simple formatted data reports, followed by a list of variables, which will be specific to the type of data being reported.

The fields in the XML Simple header block are as follows:

- NetworkInterface – 16-digit hex MAC Address of EAGLE-200 ZigBee radio that is receiving the data.

- HardwareAddress – 16-digit hex MAC Address of subdevice (usually a meter) that is reporting the data.
- Protocol – Text string describing the source of the data (usually 'Zigbee')
- Manufacturer – Text string describing the subdevice manufacturer. This must match exactly the `<Manufacturer>` field in the Device Profile for that subdevice.
- ModelId – Text string describing the subdevice model. This must match exactly the `<ModelId>` field in the Device Profile for that subdevice.
- TimeStamp – Integer with the number of seconds since Jan.1, 1970.
- ComponentFixedId – Integer that identifies the component within subdevice that the data is coming from (usually "0" for a meter).
- MainTag – Text that describes the type of data being reported
- Variables – Tag that indicates the start of the list of variables. The list ends with the `</Variables>` tag.

Each variable is enclosed by the `<variable>` and `</variable>` tags, and contains the following three elements:

- Name
- Value
- Units

For brevity, the listings of data reports that follow will only show the `<MainTag>` field and variable list for the XML Simple format.

3.3. JSON Format

3.3.1. Request

If the `<Format>` field in the Configuration File is set to `RFA`, then the EAGLE-200 will upload data in JSON format. In this case, the POST requests are the same as shown above, except the body of the POST requests will have the following structure:

```
{
  "timestamp":"1474484326000",
  "deviceGuid":"d8d5b900355a",
  "body": [
  ]
}
```

Unlike XML, JSON has no Prolog. The Root Element is enclosed by curly brackets: { }. It has three attributes:

- timestamp -- this is an integer with the number of <u>milliseconds</u> (not seconds, as it is for XML) since Jan.1, 1970.
- deviceGuid – this is a 12-digit hexadecimal number. It is the MAC Address of the EAGLE-200.
- body – this is the Root Element body, which is enclosed in square brackets: [ ].

The listings of data reports that follow will only show the `"body"` portion of the JSON format for brevity.

4. Data Reports

4.1. DeviceInfo

The DeviceInfo report provides some basic information about the EAGLE-200 device.

4.1.1. XML Raw format

```
<DeviceInfo>
  <DeviceMacId>0xd8d5b9000000b200</DeviceMacId>
  <InstallCode>0xcc15a871ce590351</InstallCode>
  <LinkKey>0xe13e7838b6322f2fca2dcd9c4cf205d8</LinkKey>
  <FWVersion>2.1.6 (9463)</FWVersion>
  <HWVersion>1.3.4</HWVersion>
  <ImageType>0x2101</ImageType>
  <Manufacturer>Rainforest Automation, Inc.</Manufacturer>
  <ModelId>Z114-EAGLE3</ModelId>
  <DateCode>2017040529021003</DateCode>
  <Protocol>Zigbee</Protocol>
</DeviceInfo>
```

Element	Type	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
InstallCode	16 hex digits	Install Code for EAGLE-200 ZigBee radio
LinkKey	32 hex digits	ZigBee radio Link Key
FWVersion	Text	Firmware Version
HWVersion	Text	Hardware Version
ImageType	4 hex digits	ZigBee code image type
Manufacturer	Text	“Rainforest Automation, Inc.”
ModelId	Text	“Z114-EAGLE3”
DateCode	YYYYMMDDZZZZZZZZ	Manufacturer’s date code and lot number
Protocol	Text	Source of information (“Zigbee”)

4.1.2. XML Simple format

If `<Format>` is set to `XML:Simple`, DeviceInfo reports are not uploaded.

4.1.3. JSON format

If `<Format>` is set to `RFA`, DeviceInfo reports are not uploaded.

4.2. ConnectionStatus

The ConnectionStatus report provides detailed information about the ZigBee network that the EAGLE-200 is connected to.

4.2.1. XML Raw format

```
<ConnectionStatus>
  <DeviceMacId>0xd8d5b9000000b200</DeviceMacId>
  <MeterMacId>0x000781000081fd0b</MeterMacId>
  <Status>Rejoining</Status>
  <ExtPanId>0x000781000081fd0b</ExtPanId>
  <Channel>14</Channel>
  <ShortAddr>0xd291</ShortAddr>
  <LinkStrength>0x00</LinkStrength>
  <Protocol>Zigbee</Protocol>
</ConnectionStatus>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
MeterMacId	16 hex digits	MAC Address of Meter
Status	Initializing | Network Discovery | Joining | Join: Fail | Join: Success | Authenticating | Authenticating: Success | Authenticating: Fail | Connected | Disconnected | Rejoining	Indicates the current state of the EAGLE-200 ZigBee endpoint radio.
ExtPanId	16 hex digits	Extended PAN ID of the ZigBee network
Channel	11-26	The Zigbee radio channel on which the EAGLE-200 is operating.
ShortAddr	4 hex digits	The short network address assigned to the EAGLE-200 by the coordinator (meter).
LinkStrength	2 hex digits	Indicates the strength of the Zigbee radio link.
Protocol	Text	Source of information (“Zigbee”)

4.2.2. XML Simple format

If `<Format>` is set to `XML:Simple`, ConnectionStatus reports are not uploaded.

4.2.3. JSON format

If `<Format>` is set to `RFA`, ConnectionStatus reports are not uploaded.

4.3. TimeCluster

The TimeCluster report provides time information supplied by the meter to the EAGLE-200.

4.3.1. XML Raw format

```
<TimeCluster>
  <DeviceMacId>0xd8d5b9000000b1ff</DeviceMacId>
  <UTCTime>0x20f2d7ed</UTCTime>
  <LocalTime>0x20f2d7ed</LocalTime>
  <UTCTimeString>Fri Jul 7, 2017 11:38:21 pm</UTCTimeString>
  <LocalTimeString>Fri Jul 7, 2017 11:38:21 pm</LocalTimeString>
  <Protocol>Zigbee</Protocol>
</TimeCluster>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
UTCTime	8 hex digits	UTC Time offset in seconds from 00:00:00 01Jan2000.
LocalTime	8 hex digits	Local Time offset in seconds from 00:00:00 01Jan2000.
UTCTimeString	Text	Text version of UTCTime
LocalTimeString	Text	Text version of LocalTime
Protocol	Text	Source of information (“Zigbee”)

4.3.2. XML Simple format

If `<Format>` is set to `XML:Simple`, TimeCluster reports are not uploaded.

4.3.3. JSON format

If `<Format>` is set to `RFA`, TimeCluster reports are not uploaded.

4.4. InstantaneousDemand

InstantaneousDemand reports indicate the power (watts) being consumed at a specific point in time.

4.4.1. XML Raw format

4.4.1.1. Meter data

```
<InstantaneousDemand>
  <DeviceMacId>0xd8d5b9000000b74d</DeviceMacId>
  <MeterMacId>0x001d230100402d72</MeterMacId>
  <TimeStamp>0x211cc7a8</TimeStamp>
  <Demand>0x000032</Demand>
  <Multiplier>0x00000001</Multiplier>
  <Divisor>0x000003e8</Divisor>
  <UnitOfMeasure>0x00</UnitOfMeasure>
  <DigitsRight>0x03</DigitsRight>
  <DigitsLeft>0x06</DigitsLeft>
  <SuppressLeadingZero>Y</SuppressLeadingZero>
  <Protocol>Zigbee</Protocol>
</InstantaneousDemand>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
MeterMacId	16 hex digits	MAC Address of meter
TimeStamp	8 hex digits	UTC Time offset in seconds from 00:00:00 01Jan2000 when demand data was received from meter.
Demand	8 hex digits	The raw value of the instantaneous demand for electricity, as measured by the meter at the time.
Multiplier	8 hex digits	Multiply raw value by to get actual; if zero, use 1
Divisor	8 hex digits	Divide raw value by to get actual; if zero, use 1
UnitOfMeasure	2 hex digits	Should be 0x00, indicating demand in kW, binary format.
DigitsRight	2 hex digits	Number of digits to the right of the decimal point to display
DigitsLeft	2 hex digits	Number of digits to the left of the decimal point to display
Suppress LeadingZero	Y | N	Y: Do not display leading zeros N: Display leading zeros
Protocol	Text	Source of information (“Zigbee”)

4.4.1.2. Subdevice data

```
<ReadAttributesResponse>
  <DeviceMacId>0xd8d5b9000000b6d4</DeviceMacId>
  <TimeStamp>0x20e99d5f</TimeStamp>
  <NetworkAddress>0x8ac8</NetworkAddress>
  <IeeeAddress>0x0024460000068d1c</IeeeAddress>
  <ClusterId>0x0702</ClusterId>
  <SourceEndpoint>0x0c</SourceEndpoint>
  <DestinationEndpoint>0x01</DestinationEndpoint>
  <Attribute1>0x0400</Attribute1>
  <Value1>0x00000000</Value1>
  <Attribute2>0x0300</Attribute2>
  <Value2>0x00</Value2>
  <Attribute3>0x0301</Attribute3>
  <Value3>0x00000001</Value3>
  <Attribute4>0x0302</Attribute4>
  <Value4>0x000003e8</Value4>
  <Attribute5>0x0304</Attribute5>
  <Value5>0x3b</Value5>
  <Protocol>Zigbee</Protocol>
</ReadAttributesResponse>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
TimeStamp	8 hex digits	UTC Time offset in seconds from 00:00:00 01Jan2000 when demand data was received from the subdevice.
NetworkAddress	4 hex digits	Short Zigbee network address of subdevice.
IeeeAddress	16 hex digits	MAC Address of subdevice ZigBee radio
ClusterId	4 hex digits	Zigbee Cluster ID
SourceEndpoint	2 hex digits	ID of subdevice endpoint providing data
DestinationEndpoint	2 hex digits	ID of EAGLE-200 endpoint
AttributeX	4 hex digits	Zigbee attribute ID. Number of attributes depends upon the cluster.
ValueX	8 hex digits	Value of the corresponding attribute.
Protocol	Text	Source of information (“Zigbee”)

4.4.2. XML Simple format

```
<MainTag>InstantaneousDemand</MainTag>
 <Variables>
   <Variable>
     <Name>InstantaneousDemand</Name>
     <Value>0.000000</Value>
     <Units>kW</Units>
   </Variable>
   <Variable>
     <Name>Multiplier</Name>
     <Value>50</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>Divisor</Name>
     <Value>1000</Value>
     <Units></Units>
   </Variable>
 </Variables>
```

Variable	Type	Description
InstantaneousDemand	32-bit floating point	The raw value of the power measured at that time
Multiplier	Integer	To multiply raw value by to get actual.
Divisor	Integer	To divide raw value by to get actual.

4.4.3. JSON format

```
"timestamp":"0",
"subdeviceGuid":"ffffffffffffffff",
"componentId":"00",
"dataType":"InstantaneousDemand",
"data":{
  "demand":0.0,
  "units":"kW"
 }
```

Element	Type/Range	Description
timestamp	integer	Milliseconds in Unix time (since Jan 1, 1970)
subdeviceGuid	16 hex digits	MAC Address of the attached device that is measuring the reading (usually the meter MAC ID).
componentId	string	Identifies the component within subdevice that the data is coming from (usually “Main” for a meter).
dataType	string	Identifies this as demand data
data		List of attribute-value pairs
demand	32-bit floating point	Power being used/generated at a moment in time
units	string	Units of the power reading (usually kW)

4.5. CurrentSummation

CurrentSummation reports indicate the total, accumulated energy (kWh) consumed by an endpoint since it was first enabled. The number continuously increases unless the device is reset to factory conditions, at which point it will start at zero.

4.5.1. XML Raw format

```
<CurrentSummation>
  <DeviceMacId>0xd8d5b9000000af85</DeviceMacId>
  <MeterMacId>0xd8d5b900000021a7</MeterMacId>
  <TimeStamp>0x20acaec0</TimeStamp>
  <SummationDelivered>0x000000000001f81f</SummationDelivered>
  <SummationReceived>0x0000000000000000</SummationReceived>
  <Multiplier>0x00000001</Multiplier>
  <Divisor>0x000003e8</Divisor>
  <UnitOfMeasure>0x00</UnitOfMeasure>
  <DigitsRight>0x03</DigitsRight>
  <DigitsLeft>0x00</DigitsLeft>
  <SuppressLeadingZero>Y</SuppressLeadingZero>
  <Protocol>Zigbee</Protocol>
</CurrentSummation>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
MeterMacId	16 hex digits	MAC Address of meter
TimeStamp	8 hex digits	UTC Time (offset in seconds from 00:00:00 01Jan2000) when data received from meter.
SummationDelivered	8 hex digits	Raw value of the total summation of energy delivered from the utility to the user.
SummationReceived	8 hex digits	Raw value of the total summation of energy received by the utility from the user.
Multiplier	8 hex digits	Multiply raw value by to get actual; if zero, use 1
Divisor	8 hex digits	Divide raw value by to get actual; if zero, use 1
UnitOfMeasure	2 hex digits	Should be 0x00, indicating summation in kWh, binary format.
DigitsRight	2 hex digits	Number of digits to the right of the decimal point to display
DigitsLeft	2 hex digits	Number of digits to the left of the decimal point to display
SuppressLeadingZero	Y | N	Y: Do not display leading zeros N: Display leading zeros
Protocol	Text	Source of information (“Zigbee”)

4.5.2. XML Simple format

```
<MainTag>CurrentSummation</MainTag>
 <Variables>
   <Variable>
     <Name>CurrentSummationDelivered</Name>
     <Value>167.900000</Value>
     <Units>kWh</Units>
   </Variable>
   <Variable>
     <Name>CurrentSummationReceived</Name>
     <Value>0.000000</Value>
     <Units>kWh</Units>
   </Variable>
   <Variable>
     <Name>Multiplier</Name>
     <Value>50</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>Divisor</Name>
     <Value>1000</Value>
     <Units></Units>
   </Variable>
 </Variables>
```

Variable	Type	Description
CurrentSummationDelivered	32-bit floating point	Raw value of the total summation of energy delivered from the utility to the user.
CurrentSummationReceived	32-bit floating point	Raw value of the total summation of energy received by the utility from the user.
Multiplier	Integer	To multiply raw value by to get actual.
Divisor	Integer	To divide raw value by to get actual.

4.5.3. JSON format

```
"timestamp":"1565647200000",
"subdeviceGuid":"00078100005a499f",
"componentId":"all",
"dataType":"CurrentSummation",
"data":{
 "summationDelivered":167.900000,
 "units":"kWh",
 "summationReceived":0.000000
}
```

Element	Type/Range	Description
timestamp	integer	Milliseconds in Unix time (since Jan 1, 1970)
subdeviceGuid	16 hex digits	MAC Address of the attached device that is measuring the reading (usually the meter MAC ID).
componentId	string	Identifies the component within subdevice that the data is coming from (usually “Main” for a meter).
dataType	string	Identifies this as summation data
data		List of attribute-value pairs
summationDelivered	32-bit floating point	Total summation of energy delivered from the utility to the user.
units	string	Units of the power reading (usually kWh)
summationReceived	32-bit floating point	Total summation of energy received by the utility from the user.

4.6. MessageCluster

The MessageCluster report provides the current Zigbee text message from the meter.

4.6.1. XML Raw format

```
<MessageCluster>
  <DeviceMacId>0xd8d5b9000000af85</DeviceMacId>
  <MeterMacId>0xd8d5b900000021a7</MeterMacId>
  <TimeStamp>0x20acaeef</TimeStamp>
  <Id>0x00000000</Id>
  <Text>Welcome to the program</Text>
  <Priority>Low</Priority>
  <StartTime>0x20acad0d</StartTime>
  <Duration>0xffff</Duration>
  <ConfirmationRequired>N</ConfirmationRequired>
  <Confirmed>N</Confirmed>
  <Queue>Active</Queue>
  <Protocol>Zigbee</Protocol>
</MessageCluster>
```

Element	Type/Range	Description
DeviceMacId	16 hex digits	MAC Address of EAGLE-200 ZigBee radio
MeterMacId	16 hex digits	MAC Address of Meter
TimeStamp	8 hex digits	UTC Time (offset in seconds from 00:00:00 01Jan2000) when message received.
Id	8 hex digits	Unique Message ID
Text	Text	Contents of message, HTML encoded, with escape codes for >, <, &, “.
Priority	Low | Medium | High | Critical	Message priority
StartTime	8 hex digits	UTC Time at which message becomes valid. Zero means now.
Duration	4 hex digits	Length of time, in minutes, that message is valid.
Confirmation Required	Y | N	Y: a user confirmation is required; N: a user confirmation is not required (default)
Confirmed	Y | N	Y: user confirmation has been sent; N: user confirmation has not been sent (default)
Queue	Active | Cancel Pending	Active: message in active queue Cancel Pending: message in cancel pending queue

4.6.2. XML Simple format

```
<MainTag>MessageCluster</MainTag>
 <Variables>
   <Variable>
     <Name>MessageId</Name>
     <Value>1</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>Message</Name>
     <Value>METER_TEST12345</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>MessagePriority</Name>
     <Value>0</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>MessageStartTime</Name>
     <Value>618959194</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>MessageDurationInMinutes</Name>
     <Value>65535</Value>
     <Units>min</Units>
   </Variable>
   <Variable>
     <Name>MessageConfirmed</Name>
     <Value></Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>MessageConfirmationRequired</Name>
     <Value>false</Value>
     <Units></Units>
   </Variable>
 </Variables>
```

Element	Type/Range	Description
MessageId	Integer	Unique Message ID
Message	Text	Contents of message, HTML encoded, with escape codes for >, <, &, “.
MessagePriority	Integer	Message priority (0-3) 0=Low,1=Medium,2=High,3=Critical
MessageStartTime	Integer	UTC Time (offset in seconds from 00:00:00

		01Jan2000) at which message becomes valid. Zero means now.
MessageDuration InMinutes	Integer	Length of time, in minutes, that message is valid.
MessageConfirmed	y | n	y: user confirmation has been sent; n: user confirmation has not been sent (default)
MessageConfirmation Required	true | false	true: confirmation is required; false: confirmation is not required (default)

4.6.3. JSON format

```
"timestamp":"1565647210000",
"subdeviceGuid":"00078100005a499f",
"componentId":"all",
"dataType":"MessageCluster",
"data":{
 "MessageId":1,
 "Message":"METER_TEST12345",
 "MessagePriority":0,
 "MessageStartTime":618959194,
 "MessageDurationInMinutes":65535,
 "units":"min",
 "MessageConfirmed":"",
 "MessageConfirmationRequired":"false"
}
```

Element	Type/Range	Description
timestamp	integer	Milliseconds in Unix time (since Jan 1, 1970)
subdeviceGuid	16 hex digits	MAC Address of the attached device that is measuring the reading (usually the meter MAC ID).
componentId	string	Identifies the component within subdevice that the data is coming from (usually “Main” for a meter).
dataType	string	Identifies this as messaging data
data		List of attribute-value pairs
MessageId	integer	Unique Message ID
Message	string	Contents of message, HTML encoded, with escape codes for >, <, &, “.
MessagePriority	integer	Message priority (0-3) 0=Low,1=Medium,2=High,3=Critical
MessageStartTime	integer	UTC Time (offset in seconds from 00:00:00 01Jan2000) at which message becomes valid. Zero means now.
MessageDuration InMinutes	integer	Length of time, in minutes, that message is valid.
units	String	
MessageConfirmed	y | n	y: user confirmation has been sent; n: user confirmation has not been sent (default)
MessageConfirmation Required	true | false	true: confirmation is required; false: confirmation is not required (default)

4.7. PriceCluster

PriceCluster reports provide any pricing information that may have been loaded into the meter.

4.7.1. XML Raw format

```
<PriceCluster>
 <DeviceMacId>0xd8d5b9000000fca8</DeviceMacId>
 <MeterMacId>0x00078100005a499f</MeterMacId>
 <TimeStamp>0x24e5ffd8</TimeStamp>
 <Price>0x00000005</Price>
 <Currency>0x0348</Currency>
 <TrailingDigits>0x02</TrailingDigits>
 <Tier>0x01</Tier>
 <StartTime>0x24e60010</StartTime>
 <Duration>0xffff</Duration>
 <RateLabel>Price1</RateLabel>
 <Protocol>Zigbee</Protocol>
</PriceCluster>
```

Element	Type	Description
DeviceMacId	16 hex digits	MAC Address of **EAGLE**™ ZigBee radio
MeterMacId	16 hex digits	MAC Address of Meter
TimeStamp	8 hex digits	UTC Time (offset in seconds from 00:00:00 01Jan2000) when price data was received from meter or set by user.
Price	8 hex digits	Price from meter or set by user; will be zero if no price is set
Currency	4 hex digits	Currency being used; value of this field matches the values defined by ISO 4217
TrailingDigits	2 hex digits	The number of implicit decimal places in the price. (e.g. 2 means divide Price by 100).
Tier	2 hex digits	The price Tier in effect (1-5).
StartTime	8 hex digits	UTC Time at which price becomes valid. Zero means now.
Duration	4 hex digits	Length of time, in minutes, that price is valid.
RateLabel	Text	Rate label for the current price tier; will be “Set by User” if a user-defined price is set
Protocol	Text	Source of information (“Zigbee”)

4.7.2. XML Simple format

```
<MainTag>PriceCluster</MainTag>
 <Variables>
   <Variable>
     <Name>Price</Name>
     <Value>0.050000</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>PriceCurrency</Name>
     <Value>USD</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>PriceTrailingDigits</Name>
     <Value>2</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>PriceTier</Name>
     <Value>1</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>PriceStartTime</Name>
     <Value>1565646751</Value>
     <Units></Units>
   </Variable>
   <Variable>
     <Name>PriceDuration</Name>
     <Value>65535</Value>
     <Units>min</Units>
   </Variable>
   <Variable>
     <Name>PriceRateLabel</Name>
     <Value>Price1</Value>
     <Units></Units>
   </Variable>
 </Variables>
```

Element	Type	Description
Price	32-bit floating point	Price from meter or set by user; will be zero if no price is set
PriceCurrency	Text	Currency being used; this field uses the codes defined by ISO 4217
PriceTrailingDigits	Integer	This can be ignored, as it has already been incorporated into the Price variable.

PriceTier	Integer	The price Tier in effect (1-5).
PriceStartTime	Integer	UTC Time (offset in seconds from 00:00:00 01Jan2000) at which price becomes valid. Zero means now.
PriceDuration	Integer	Length of time, in minutes, that price is valid.
PriceRateLabel	Text	Rate label for the current price tier; will be “Set by User” if a user-defined price is set

4.7.3. JSON format

```
"timestamp":"1565798230000",
"subdeviceGuid":"00078100005a499f",
"componentId":"all",
"dataType":"Price",
"data":{
 "price":0.050000,
 "PriceCurrency":"USD",
 "PriceTrailingDigits":2,
 "PriceTier":1,
 "PriceStartTime":1565798230,
 "PriceDuration":65535,
 "units":"min",
 "PriceRateLabel":"Price1"
}
```

Element	Type	Description
timestamp	integer	Milliseconds in Unix time (since Jan 1, 1970)
subdeviceGuid	16 hex digits	MAC Address of the attached device that is measuring the reading (usually the meter MAC ID).
componentId	string	Identifies the component within subdevice that the data is coming from (usually “Main” for a meter).
dataType	string	Identifies this as messaging data
data		List of attribute-value pairs
price	32-bit floating point	Price from meter or set by user; will be zero if no price is set
PriceCurrency	string	Currency being used; this field uses the codes defined by ISO 4217
PriceTrailingDigits	integer	This can be ignored, as it has already been incorporated into the price number.
PriceTier	integer	The price Tier in effect (1-5).
PriceStartTime	integer	UTC Time (offset in seconds from 00:00:00 01Jan2000) at which price becomes valid. Zero means now.
PriceDuration	integer	Length of time, in minutes, that price is valid.
units	text	Units for the price duration (“min”).
PriceRateLabel	text	Rate label for the current price tier; will be “Set by User” if a user-defined price is set.

5. Batch Upload

The upload manager can upload data in “batch” format – saving up multiple data readings to upload in a single POST. There are a number of situations where this would happen:

- If “real time” data is not required and internet usage is at a premium – using a cellular connection, for instance – then the UploadSize and UploadPeriod elements of the Configuration file can be adjusted to queue up data readings as required to send in batch. This saves on the IP packet overhead for each reading.
- Batch data can also be sent if the network connection between the EAGLE-200 and the cloud server goes down for a period of time. As long as the Uploader is not in Streaming mode (i.e., UploadSize > 0), then data readings will queue up and all be sent at once when the network connection comes back up.
- There may also be inadvertent batch uploads when two pieces of data are ready at the same time, such as occasional status updates ‘piggybacking’ on regular meter readings. For this reason, it is important to design the parser in the server interface to the Uploader API to expect multiple Data Reports in a POST and to discard any irrelevant data.

In the case of a batch upload, the body of the Root Element of the POST will consist of a concatenated series of Data Reports – all of the same format. In the case of JSON formatted data, the individual Data Reports are separated by a comma.

<u>Example</u> – XML Raw format:

```
<?xml version="1.0"?>
<rainforest timestamp="0000000000s" version="2.0"
macId="0xffffffffffff">
<InstantaneousDemand>
<DeviceMacId>0xd8d5b9000000b74d</DeviceMacId>
<MeterMacId>0x001d230100402d72</MeterMacId>
<TimeStamp>0x211cc7a8</TimeStamp>
<Demand>0x000032</Demand>
<Multiplier>0x00000001</Multiplier>
<Divisor>0x000003e8</Divisor>
<UnitOfMeasure>0x00</UnitOfMeasure>
<DigitsRight>0x03</DigitsRight>
<DigitsLeft>0x06</DigitsLeft>
<SuppressLeadingZero>Y</SuppressLeadingZero>
<Protocol>Zigbee</Protocol>
</InstantaneousDemand>
<CurrentSummation>
<DeviceMacId>0xd8d5b9000000af85</DeviceMacId>
<MeterMacId>0xd8d5b900000021a7</MeterMacId>
<TimeStamp>0x20acaec0</TimeStamp>
<SummationDelivered>0x000000000001f81f</SummationDelivered>
<SummationReceived>0x0000000000000000</SummationReceived>
<Multiplier>0x00000001</Multiplier>
<Divisor>0x000003e8</Divisor>
<UnitOfMeasure>0x00</UnitOfMeasure>
```

```
<DigitsRight>0x03</DigitsRight>
<DigitsLeft>0x00</DigitsLeft>
<SuppressLeadingZero>Y</SuppressLeadingZero>
<Protocol>Zigbee</Protocol>
</CurrentSummation>
</rainForest>
```

<u>Example</u> – JSON format:

```
{
  "timestamp":"1474484326000",
  "deviceGuid":"d8d5b900355a",
  "body": [{
  "timestamp":"1474484240000",
  "subdeviceGuid":"001bc5007200578f",
  "componentId":"01",
  "dataType":"InstantaneousDemand",
  "data":{
    "demand":2.0,
    "units":"kW"
  }
},{
  "timestamp":"1474484280100",
  "subdeviceGuid":"001bc5007200578f",
  "componentId":"01",
  "dataType":"CurrentSummation",
  "data":{
    "summationDelivered":0.278,
    "summationReceived":0.69,
    "units":"kWh"
  }}
  ]
}
```

6. Uploader Configuration

Each EAGLE-200 can support two active Uploaders. One will point to the Rainforest Cloud, while the other can be user-defined and can point to any URL. The user-defined Uploader is defined by its associated configuration file, as described in section 1.1.2 of this document.

6.1. Creating a Configuration File

A user-defined configuration file can be created in two ways:

- manually, using the online web User Portal, or
- through the Rainforest Cloud REST API.

6.1.1. Manual

To create a configuration file manually, log in to your Rainforest Cloud Account and select the EAGLE-200 that you want to configure from the dropdown "Cloud ID" list (if the EAGLE-200 does not appear on your accounts' list, you can add it from the EAGLEs tab in the Settings page). Go to the Settings page and select the Cloud tab. Under the "Add Upload Destination" heading you will see a dropdown list labelled "Select Destination". This list has a number of pre-defined Uploaders to send data to third-party cloud service providers. Selecting one of these will cause the corresponding configuration file to apply to the second Uploader. The first item in the dropdown list is "Custom"; selecting this will expand the page to allow you to define your own configuration file. There are fields to enter Label ("name" element), Protocol, Hostname, URL, Port, Username (optional), Password (optional), Format, which all correspond to elements in the configuration file. Clicking the "Add" button will create a new configuration file with the elements you specified and will add the Label to the "Select Destination" dropdown list.

6.1.2. REST API

To create a configuration file using the Rainforest Cloud REST API, you should send an HTTPS POST with the following structure:

```
POST /rest/device/d8d5b9xxxxxx/command HTTP/1.1
Host: api.rainforestcloud.com
Content-type: application/json
Content-Length: xxx
Authorization: Basic xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

{
"command": "uploader",
"parameters": {
"name": "uploader_add",
"description": "custom uploader for XYZ corp",
"provider": "XYZ",
"format": "XML:RAW",
"hostname": "api.xyz.com",
"port": 80,
"url": "/meter/data",
"uploadSize": 2,
"uploadPeriod": 60,
"protocol": "https",
"x-header1": "",
"x-header2": "",
"x-header3": ""
}
}
```

Where:

- `xxxxxx` is the Cloud ID of the target EAGLE-200
- `xxx` is the number of characters in the body of the POST
- `xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx` is the 32-character Basic Authentication credential formed using the EAGLE-200 Cloud ID as the username, and the EAGLE-200 Install Code as the password.
- `"name":` is the command name; in this case, `"uploader_add"`.
- The rest of the parameters are elements of the configuration file, as described in section 1.1.2 of this document.

Successful responses to the POST will have the code 200 or 201.

Failed response codes:

- 409: There is already an uploader with the same "provider" string.
- 403: Only an administrator can perform that action.
- 500: Internal error. There is nothing further that the user can do.

Rainforest Cloud REST API commands can also be issued directly from our Swagger page: https://api.rainforestcloud.com/swagger.
This site contains extensive details about the REST API.

Note that once a Configuration File is created, its elements cannot be changed. If you want to change elements of an existing Configuration File, you will have to delete the existing uploader and its associated Configuration File, and then recreate it with the new element values.

To delete the uploader, issue the `uploader_delete` command:

```
{
"command": "uploader",
"parameters": {
"name": "uploader_delete",
"provider": "XYZ",
}
}
```

6.2. Adjusting the Meter Polling

The EAGLE-200 Uploaders forward data that is received from the internal Zigbee radio to the cloud. The Zigbee radio collects data from the meter by asking (polling) for it at regular intervals defined by a schedule. This schedule determines what type, how much, and how often data is uploaded.

The Rainforest Cloud REST API can be used to adjust the polling schedule. To do this, use the same POST structure as shown above, except the body of the post will look like:

```
{
"command": "set_schedule",
"parameters": {
"subdeviceGuid": "0000000000000000",
"event": "summation",
"frequency": 5,
"mode": "default",
"enabled": true
}
}
```

Where:

- `"subdeviceGuid"`: 16-character string, MAC Address of the EAGLE-200 Zigbee radio.
- `"event"`: a string which is one of the following meter data types:
    - `"summation"`
    - `"demand"`
    - `"price"`
- `"frequency"`: an integer that defines the polling rate, in seconds.
- `"mode"`: should always be `"default"`.
- `"enabled"`: a Boolean value (`true/false`) used to turn polling on or off for the event.